WHAT IS CLAIMED IS:

1. A processing core that executes a parallel multiply accumulate operation, the processing core comprising:

a first input operand register comprising a plurality of first input operands;

a second input operand register comprising a plurality of second input operands;

a third input operand register comprising a plurality of third input operands;

a plurality of functional blocks that each perform a multiply accumulate operation;

an output operand register comprising a plurality of output operands, wherein each of the plurality of output operands is related to one of the plurality of first input operands, one of the plurality of second input operands and one of the plurality of third input operands.

2. The processing core as set forth in claim 1, wherein each of the plurality of functional blocks produce a result equal to multiplying the first input operand with the second input operand to produce a product and adding the third input operand to that product.

3. The processing core as set forth in claim 2, wherein the product is truncated before adding the product to the third input operand.

4. The processing core as set forth in claim 2, wherein the product is rounded before adding the product to the third input operand.

5. The processing core as set forth in claim 1, wherein a very long instruction word includes a plurality of parallel multiply accumulate operations.

6. The processing core as set forth in claim 1, wherein the first through third input operands are at least one of following formats: integer, floating-point, fixed-point, and two's complement.

7. The processing core as set forth in claim 1, wherein the first through third input operands are at least one of: 128 bit values, 64 bit values, 32 bit values, 16 bit values, and 8 bit values.

8. A method for determining a plurality of output operands, the method comprising:

loading a multiply accumulate instruction;

decoding the multiply accumulate instruction;

loading a plurality of multiplicands, a plurality of multipliers and a plurality of accumulate values;

processing a first multiplicand, a first multiplier and a first accumulated value in order to produce a first output operand;

processing a second multiplicand, a second multiplier and a second accumulated value in order to produce a second output operand; and

storing the first output operand and second output operand.

9. The method of claim 8, further comprising:

loading a second multiply accumulate instruction;

decoding the second multiply accumulate instruction;

loading a second plurality of multiplicands, a second plurality of multipliers and a second plurality of accumulate values;

processing a third multiplicand, a third multiplier and a third accumulated value in order to produce a third output operand;

processing a fourth multiplicand, a fourth multiplier and a fourth accumulated value in order to produce a fourth output operand;

storing the third output operand and fourth output operand; and

concatenating the two of the first through fourth output operands together to form a larger output operand.

10. The method of claim 8, wherein the multiply accumulate instruction and the second multiply accumulate instruction are included in the same very long instruction word (VLIW).

11. The method of claim 8, further comprising:

determining if the first output operand is larger than a predetermined value; and

replacing the first output operand with a predetermined constant.

12. The method of claim 11, wherein the predetermined value is equal to a largest or smallest value for the first and second output operand.

13. A method for determining a plurality of output operands, the method comprising:

loading a multiply accumulate instruction;

decoding the multiply accumulate instruction;

loading a plurality of multiplicands, a plurality of multipliers and a plurality of accumulate values;

determining a first result equal to a product of a first multiplicand and a first multiplier plus a first accumulate value; and

storing the first result.

14. The method of claim 13, further comprising:

loading a second multiply accumulate instruction;

decoding the second multiply accumulate instruction;

loading a second plurality of multiplicands, a second plurality of multipliers and a second plurality of accumulate values;

determining a second result equal to a product of a second multiplicand and a second multiplier plus a second accumulate value;

storing the second result; and

arranging the first result and second result into an unified result occupying more bits than either the first or second results.

15. The method of claim 13, wherein the multiply accumulate instruction and the second multiply accumulate instruction are included in the same very long instruction word (VLIW).

16. The method of claim 13, further comprising:

determining a second result equal to a product of a second multiplicand and a second multiplier plus a second accumulate value; and

storing the second result.

17. The method of claim 13, further comprising:

determining if the first result is larger than a predetermined value; and

replacing the first result with a predetermined constant.

18. The method of claim 17, wherein the predetermined value is equal to a largest or smallest value for the first and second output operand.